

# Efficient Identity-Based and Authenticated Key Agreement Protocol

Yongge Wang

UNC Charlotte (USA), yonwang@uncc.edu

**Abstract.** Several identity based and implicitly authenticated key agreement protocols have been proposed in recent years and none of them has achieved all required security properties. In this paper, we propose an efficient identity-based and authenticated key agreement protocol IDAK using Weil/Tate pairing. The security of IDAK is proved in Bellare-Rogaway model. Several required properties for key agreement protocols are not implied by the Bellare-Rogaway model. We proved these properties for IDAK separately.

## 1 Introduction

Key establishment protocols are one of the most important cryptographic primitives that have been used in our society. The first unauthenticated key agreement protocol based on asymmetric cryptographic techniques were proposed by Diffie and Hellman [15]. Since this seminal result, many authenticated key agreement protocols have been proposed and the security properties of key agreement protocols have been extensively studied. In order to implement these authenticated key agreement protocols, one needs to get the corresponding party's authenticated public key. For example, in order for Alice and Bob to execute the NIST recommended MQV key agreement protocol [20,26], Alice needs to get an authenticated public key  $g^b$  for Bob and Bob needs to get an authenticated public key  $g^a$  for Alice first, where  $a$  and  $b$  are Alice and Bob's private keys respectively. One potential approach for implementing these schemes is to deploy a public key infrastructure (PKI) system, which has proven to be difficult. Thus it is preferred to design easy to deploy authenticated key agreement systems. Identity based key agreement system is such an example.

In 1984, Shamir [32] proposed identity based cryptosystems where user's identities (such as email address, phone numbers, office locations, etc.) could be used as the public keys. Several identity based key agreement protocols (see, e.g., [11,17,22,27,30,31,33,36,38]) have been proposed since then. Most of them are not practical or do not have all required security properties. Joux [18] proposed a one-round tripartite non-identity based key agreement protocol using Weil pairing. Then feasible identity based encryption schemes based on Weil or Tate pairing were introduced by Sakai, Ohgishi, and Kasahara [30] and later by Boneh and Franklin [7] independently.

Based on Weil and Tate pairing techniques, Smart [36], Chen-Kudla [11], Scott [31], Shim [33], and McCullagh-Barreto [22] designed identity based and authenticated key agreement protocols. Chen-Kudla [11] showed that Smart's protocol is not secure in several aspects. Cheng et al. [13] pointed out that Chen-Kudla's protocol is not secure

against unknown key share attacks. Scott's protocol is not secure against man in the middle attacks. Sun and Hsieh [37] showed that Shim's protocol is insecure against key compromise impersonation attacks or man in the middle attacks. Choo [14] showed that McCullagh and Barreto's protocol is insecure against key revealing attacks. McCullagh and Barreto [23] revised their protocol. But the revised protocol does not achieve weak perfect forward secrecy property. In this paper, we propose an efficient identity based and authenticated key agreement protocol achieving all security properties that an authenticated key agreement protocol should have.

The advantage of identity based key agreement is that non-PKI system is required. The only prerequisite for executing identity based key agreement protocols is the deployment of authenticated system-wide parameters. Thus, it is easy to implement these protocols in relatively closed environments such as government organizations and commercial entities.

The remainder of this paper is organized as follows. In §2 we briefly describe bilinear maps, bilinear Diffie-Hellman problem, and its variants. In §3, we describe our identity based and authenticated key agreement protocol IDAK. §4 describes a security model for identity based key agreement. In section §5, we prove the security of IDAK key agreement protocol. In sections §6 and §7, we discuss key compromise impersonation resilience and perfect forward secrecy properties of IDAK key agreement protocol.

## 2 Bilinear maps and the bilinear Diffie-Hellman assumptions

In the following, we briefly describe the bilinear maps and bilinear map groups. The details could be found in Joux [18] and Boneh and Franklin [7].

1.  $G$  and  $G_1$  are two (multiplicative) cyclic groups of prime order  $q$ .
2.  $g$  is a generator of  $G$ .
3.  $\hat{e} : G \times G \rightarrow G_1$  is a bilinear map.

A bilinear map is a map  $\hat{e} : G \times G \rightarrow G_1$  with the following properties:

1. bilinear: for all  $g_1, g_2 \in G$ , and  $x, y \in \mathbb{Z}$ , we have  $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$ .
2. non-degenerate:  $\hat{e}(g, g) \neq 1$ .

We say that  $G$  is a bilinear group if the group action in  $G$  can be computed efficiently and there exists a group  $G_1$  and an efficiently computable bilinear map  $\hat{e} : G \times G \rightarrow G_1$  as above. Concrete examples of bilinear groups are given in [18,7]. For convenience, throughout the paper, we view both  $G$  and  $G_1$  as multiplicative groups though the concrete implementation of  $G$  could be additive elliptic curve groups.

Throughout the paper *efficient* means probabilistic polynomial-time, *negligible* refers to a function  $\varepsilon_k$  which is smaller than  $1/k^c$  for all  $c > 0$  and sufficiently large  $k$ , and *overwhelming* refers to a function  $1 - \varepsilon_k$  for some negligible  $\varepsilon_k$ . Consequently, a function  $\delta_k$  is *non-negligible* if there exists a constant  $c$  and there are infinitely many  $k$  such that  $\delta_k > 1/k^c$ . We first formally define the notion of a bilinear group family and computational indistinguishable distributions (some of our terminologies are adapted from Boneh [6]).

**Bilinear group families** A *bilinear group family*  $\mathcal{G}$  is a set  $\mathcal{G} = \{G_\rho\}$  of bilinear groups  $G_\rho = \langle G, G_1, \hat{e} \rangle$  where  $\rho$  ranges over an infinite index set,  $G$  and  $G_1$  are two groups of prime order  $q_\rho$ , and  $\hat{e} : G \times G \rightarrow G_1$  is a bilinear map. We denote by  $|\rho|$  the length of the binary representation of  $\rho$ . We assume that group and bilinear operations in  $G_\rho = \langle G, G_1, \hat{e} \rangle$  are efficient in  $|\rho|$ . Unless specified otherwise, we will abuse our notations by using  $q$  as the group order instead of  $q_\rho$  in the remaining part of this paper.

**Instance generator** An *Instance Generator*,  $\mathcal{IG}$ , for a bilinear group family  $\mathcal{G}$  is a randomized algorithm that given an integer  $k$  (in unary, that is,  $1^k$ ), runs in polynomial-time in  $k$  and outputs some random index  $\rho$  for  $G_\rho = \langle G, G_1, \hat{e} \rangle$ , and a generator  $g$  of  $G$ , where  $G$  and  $G_1$  are groups of prime order  $q$ . Note that for each  $k$ , the Instance Generator induces a distribution on the set of indices  $\rho$ .

The following Bilinear Diffie-Hellman Assumption (BDH) has been used by Boneh and Franklin [7] to show security of their identity-based encryption scheme.

**Bilinear Diffie-Hellman Problem** Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family and  $g$  be a generator for  $G$ , where  $G_\rho = \langle G, G_1, \hat{e} \rangle$ . The BDH problem in  $\mathcal{G}$  is as follows: given  $\langle g, g^x, g^y, g^z \rangle$  for some  $x, y, z \in \mathbb{Z}_q^*$ , compute  $\hat{e}(g, g)^{xyz} \in G_1$ . A CBDH algorithm  $\mathcal{C}$  for  $\mathcal{G}$  is a probabilistic polynomial-time algorithm that can compute the function  $\text{BDH}_g(g^x, g^y, g^z) = \hat{e}(g, g)^{xyz}$  in  $G_\rho$  with a non-negligible probability. That is, for some fixed  $c$  we have

$$\Pr [\mathcal{C}(\rho, g, g^x, g^y, g^z) = \hat{e}(g, g)^{xyz}] \geq \frac{1}{k^c} \quad (1)$$

where the probability is over the random choices of  $x, y, z$  in  $\mathbb{Z}_q^*$ , the index  $\rho$ , the random choice of  $g \in G$ , and the random bits of  $\mathcal{A}$ .

**CBDH Assumption.** The bilinear group family  $\mathcal{G} = \{G_\rho\}$  *satisfies* the CBDH-Assumption if there is no CBDH algorithm for  $\mathcal{G}$ . A perfect-CBDH algorithm  $\mathcal{C}$  for  $\mathcal{G}$  is a probabilistic polynomial-time algorithm that can compute the function  $\text{BDH}_g(g^x, g^y, g^z) = \hat{e}(g, g)^{xyz}$  in  $G_\rho$  with overwhelming probability.  $\mathcal{G}$  *satisfies* the perfect-CBDH-Assumption if there is no perfect-CBDH algorithm for  $\mathcal{G}$ .

**Theorem 1.** *A bilinear group family  $\mathcal{G}$  satisfies the CBDH-Assumption if and only if it satisfies the perfect-CBDH-Assumption.*

**Proof.** See Appendix. □

Consider Joux's tripartite key agreement protocol [18]: Alice, Bob, and Carol fix a bilinear group  $\langle G, G_1, \hat{e} \rangle$ . They select  $x, y, z \in_R \mathbb{Z}_q^*$  and exchange  $g^x, g^y$ , and  $g^z$ . Their shared secret is  $\hat{e}(g, g)^{xyz}$ . To *totally break* the protocol a passive eavesdropper, Eve, must compute the BDH function:  $\text{BDH}_g(g^x, g^y, g^z) = \hat{e}(g, g)^{xyz}$ .

CBDH-Assumption by itself is not sufficient to prove that Joux's protocol is useful for practical cryptographic purposes. Even though Eve may be unable to recover the entire secret, she may still be able to predict quite a few bits (less than  $c \log k$  bits for some constant  $c$ ; Otherwise, CBDH assumption is violated) of information for  $\hat{e}(g, g)^{xyz}$  with some confidence. If  $\hat{e}(g, g)^{xyz}$  is to be the basis of a shared secret key, one must bound the amount of information Eve is able to deduce about it, given  $g^x, g^y$ , and  $g^z$ . This is formally captured by the, much stronger, Decisional Bilinear Diffie-Hellman assumption (DBDH-Assumption)

**Definition 1.** Let  $\{\mathcal{X}_\rho\}$  and  $\{\mathcal{Y}_\rho\}$  be two ensembles of probability distributions, where for each  $\rho$  both  $\mathcal{X}_\rho$  and  $\mathcal{Y}_\rho$  are defined over the same domain. We say that the two ensembles are computationally indistinguishable if for any probabilistic polynomial-time algorithm  $\mathcal{D}$ , and any  $c > 0$  we have

$$|\Pr[\mathcal{D}(\mathcal{X}_\rho) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_\rho) = 1]| < \frac{1}{k^c}$$

for all sufficiently large  $k$ , where the probability is taken over all  $\mathcal{X}_\rho$ ,  $\mathcal{Y}_\rho$ , and internal coin tosses of  $\mathcal{D}$ .

In the remainder of the paper, we will say in short that the two distributions  $\mathcal{X}_\rho$  and  $\mathcal{Y}_\rho$  are computationally indistinguishable.

Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family. We consider the following two ensembles of distributions:

- $\{\mathcal{X}_\rho\}$  of random tuples  $\langle \rho, g, g^x, g^y, g^z, \hat{e}(g, g)^t \rangle$ , where  $g$  is a random generator of  $G$  ( $G_\rho = \langle G, G_1, \hat{e} \rangle$ ) and  $x, y, z, t \in_R Z_q$ .
- $\{\mathcal{Y}_\rho\}$  of tuples  $\langle \rho, g, g^x, g^y, g^z, \hat{e}(g, g)^{xyz} \rangle$ , where  $g$  is a random generator of  $G$  and  $x, y, z \in_R Z_q$ .

An algorithm that solves the Bilinear Diffie-Hellman decision problem is a polynomial time probabilistic algorithm that can effectively distinguish these two distributions. That is, given a tuple coming from one of the two distributions, it should output 0 or 1, and there should be a non-negligible difference between (a) the probability that it outputs a 1 given an input from  $\{\mathcal{X}_\rho\}$ , and (b) the probability that it outputs a 1 given an input from  $\{\mathcal{Y}_\rho\}$ . The bilinear group family  $\mathcal{G}$  satisfies the *DBDH-Assumption* if the two distributions are computationally indistinguishable.

**Remark.** The DBDH-Assumption is implied by a slightly weaker assumption: *perfect-DBDH-Assumption*. A perfect-DBDH statistical test for  $\mathcal{G}$  distinguishes the inputs from the above  $\{\mathcal{X}_\rho\}$  and  $\{\mathcal{Y}_\rho\}$  with overwhelming probability. The bilinear group family  $\mathcal{G}$  satisfies the *perfect-DBDH-Assumption* if there is no such probabilistic polynomial-time statistical test.

### 3 The scheme IDAK

In this section, we describe our identity-based and authenticated key agreement scheme IDAK. Let  $k$  be the security parameter given to the setup algorithm and  $\mathcal{IG}$  be a bilinear group parameter generator. We present the scheme by describing the three algorithms: **Setup**, **Extract**, and **Exchange**.

**Setup:** For the input  $k \in Z^+$ , the algorithm proceeds as follows:

1. Run  $\mathcal{IG}$  on  $k$  to generate a bilinear group  $G_\rho = \{G, G_1, \hat{e}\}$  and the prime order  $q$  of the two groups  $G$  and  $G_1$ .
2. Pick a random master secret  $\alpha \in Z_q^*$ .
3. Choose cryptographic hash functions  $H : \{0, 1\}^* \rightarrow G$  and  $\pi : G \times G \rightarrow Z_q^*$ . In the security analysis, we view  $H$  and  $\pi$  as random oracles. In practice, we take  $\pi$  as a random oracle (secure hash function) from  $G \times G$  to  $Z_{2^{\lceil \log q \rceil / 2}}^*$  (see Appendix for details).

The system parameter is  $\langle q, g, G, G_1, \hat{e}, H, \pi \rangle$  and the master secret key is  $\alpha$ .

**Extract:** For a given identification string  $ID \in \{0, 1\}^*$ , the algorithm computes a generator  $g_{ID} = H(ID) \in G$ , and sets the private key  $d_{ID} = g_{ID}^\alpha$  where  $\alpha$  is the master secret key.

**Exchange:** For two participants Alice and Bob whose identification strings are  $ID_A$  and  $ID_B$  respectively, the algorithm proceeds as follows.

1. Alice selects  $x \in_R Z_q^*$ , computes  $R_A = g_{ID_A}^x$ , and sends it to Bob.
2. Bob selects  $y \in_R Z_q^*$ , computes  $R_B = g_{ID_B}^y$ , and sends it to Alice.
3. Alice computes  $s_A = \pi(R_A, R_B)$ ,  $s_B = \pi(R_B, R_A)$ , and the shared secret  $sk_{AB}$  as

$$\hat{e}(g_{ID_A}, g_{ID_B})^{(x+s_A)(y+s_B)\alpha} = \hat{e}(d_{ID_A}^{(x+s_A)}, g_{ID_B}^{s_B} \cdot R_B).$$

4. Bob computes  $s_A = \pi(R_A, R_B)$ ,  $s_B = \pi(R_B, R_A)$ , and the shared secret  $sk_{BA}$  as

$$\hat{e}(g_{ID_A}, g_{ID_B})^{(x+s_A)(y+s_B)\alpha} = \hat{e}(g_{ID_A}^{s_A} \cdot R_A, d_{ID_B}^{(y+s_B)}).$$

In the next section, we will show that IDAK protocol is secure in Bellare and Rogaway [4] model with random oracle plus DBDH-Assumption. We conclude this section with a theorem which says that the shared secret established by the IDAK key agreement protocol is computationally indistinguishable from a random value.

**Theorem 2.** Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family,  $G_\rho = \langle G, G_1, \hat{e} \rangle$ , and  $g_1, g_2$  be random generators of  $G$ . Assume that DBDH-Assumption holds for  $\mathcal{G}$ . Then the distributions  $\langle g_1, g_2, g_1^x, g_2^y, \hat{e}(g_1, g_2)^{(x+\pi(g_1^x, g_2^y))(y+\pi(g_2^y, g_1^x))\alpha} \rangle$  and  $\langle g_1, g_2, g_1^x, g_2^y, \hat{e}(g_1, g_2)^z \rangle$  are computationally indistinguishable, where  $\alpha, x, y, z$  are selected from  $Z_q^*$  uniformly.

Before we give a proof for Theorem 2, we first prove two lemmas that will be used in the proof of the Theorem.

**Lemma 1.** (Naor and Reingold [24]) Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family,  $G_\rho = \langle G, G_1, \hat{e} \rangle$ ,  $m$  be a constant,  $g$  be a random generator of  $G$ , and  $\hat{g} = \hat{e}(g, g)$ . Assume that the DBDH-Assumption holds for  $G_\rho$ . Then the two distributions  $\langle \mathcal{R}, (\hat{g}^{x_i y_j z_l} : i, j, l \leq m) \rangle$  and  $\langle \mathcal{R}, (\hat{g}^{u_{ijl}} : i, j, l \leq m) \rangle$  are computationally indistinguishable. Here  $\mathcal{R}$  denotes the tuple  $(g, (g^{x_i}, g^{y_j}, g^{z_l} : i, j, l \leq m))$  and  $x_i, y_j, z_l, u_{ijl} \in_R Z_q$ .

**Proof.** Using a random reduction, Naor and Reingold [24, Lemma 4.4] (see also Shoup [35, §5.3.2] showed that the two distributions  $\langle \mathcal{R}, (g^{x_i y_j} : i, j \leq m) \rangle$  and  $\langle \mathcal{R}, (g^{u_{ij}} : i, j \leq m) \rangle$  are computationally indistinguishable. The proof can be directly modified to obtain a proof for this Lemma. The details are omitted.  $\square$

**Lemma 2.** Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family,  $G_\rho = \langle G, G_1, \hat{e} \rangle$ ,  $g$  be a random generator of  $G$ ,  $\hat{g} = \hat{e}(g, g)$ , and  $f_1$  and  $f_2$  be two polynomial-time computable functions. If the two distributions  $\mathcal{X}_1 = \langle \mathcal{R}, \hat{g}^{f_1(\mathbf{x})}, \hat{g}^{f_2(\mathbf{x})} \rangle$  and  $\mathcal{Y}_1 = \langle \mathcal{R}, \hat{g}^{z_1}, \hat{g}^{z_2} \rangle$  are computationally indistinguishable, then the two distributions  $\mathcal{X}_2 = \langle \mathcal{R}_1, \hat{g}^{f_1(\mathbf{x})+f_2(\mathbf{x})} \rangle$  and  $\mathcal{Y}_2 = \langle \mathcal{R}_2, \hat{g}^z \rangle$  are computationally indistinguishable, where  $\mathcal{R} = (g, (g^{x_i} : 1 \leq i \leq m))$ ,  $\mathbf{x} = (x_1, \dots, x_m)$ , and  $x_i, z_1, z_2, z \in_R Z_q$ .

**Proof.** See Appendix. □

**Proof of Theorem 2** Let  $\hat{g} = \hat{e}(g, g)$ . By Lemma 1, the two distributions

$$\begin{aligned}\mathcal{X} &= \langle g, g^\alpha, g^x, g^y, \hat{g}^{\alpha xy}, \hat{g}^{\alpha x \pi(g^y, g^x)}, \hat{g}^{\alpha y \pi(g^x, g^y)}, \hat{g}^{\alpha \pi(g^x, g^y) \pi(g^y, g^x)} \rangle \quad \text{and} \\ \mathcal{Y} &= \langle g, g^\alpha, g^x, g^y, \hat{g}^{z'_1}, \hat{g}^{z'_2 \pi(g^y, g^x)}, \hat{g}^{z'_3 \pi(g^x, g^y)}, \hat{g}^{z'_4 \pi(g^x, g^y) \pi(g^y, g^x)} \rangle\end{aligned}$$

are computationally indistinguishable assuming that DBDH-Assumption holds for  $\mathcal{G}$ , where  $g$  is a random generator of  $G_\rho$  and  $\alpha, x, y, z'_1, z'_2, z'_3, z'_4 \in_R Z_q$ . Since  $\pi$  is a fixed function from  $G$  to  $Z_q^*$  and  $q$  is a prime, it is straightforward to verify that for any  $\alpha, x, y \in Z_q$ ,  $\hat{g}^{z'_2 \pi(g^y, g^x)}$ ,  $\hat{g}^{z'_3 \pi(g^x, g^y)}$ , and  $\hat{g}^{z'_4 \pi(g^x, g^y) \pi(g^y, g^x)}$  are uniformly (and independently of each other) distributed over  $G_1$ . It follows that the distribution

$$\mathcal{Z} = \langle g, g^\alpha, g^x, g^y, \hat{g}^{z_1}, \hat{g}^{z_2}, \hat{g}^{z_3}, \hat{g}^{z_4} \rangle$$

is computationally indistinguishable from the distribution  $\mathcal{Y}$ , where  $z_1, z_2, z_3, z_4 \in_R Z_q$ . Thus  $\mathcal{X}$  and  $\mathcal{Z}$  are computationally indistinguishable. The Theorem now follows from Lemma 2. □

## 4 The security model

Our security model is based on Bellare and Rogaway [4] security models for key agreement protocols with several modifications. In our model, we assume that we have at most  $m \leq \text{poly}(k)$  protocol participants (principals):  $\text{ID}_1, \dots, \text{ID}_m$ , where  $k$  is the security parameter. The protocol determines how principals behave in response to input signals from their environment. Each principal may execute the protocol multiple times with the same or different partners. This is modelled by allowing each principal to have different instances that execute the protocol. An oracle  $\Pi_{i,j}^s$  models the behavior of the principal  $\text{ID}_i$  carrying out a protocol session in the belief that it is communicating with the principal  $\text{ID}_j$  for the  $s$ th time. One given instance is used only for one time. Each  $\Pi_{i,j}^s$  maintains a variable *view* (or *transcript*) consisting of the protocol run transcripts so far.

The adversary is modelled by a probabilistic polynomial time Turing machine that is assumed to have complete control over all communication links in the network and to interact with the principals via oracle accesses to  $\Pi_{i,j}^s$ . The adversary is allowed to execute any of the following queries:

- **Extract**(ID). This allows the adversary to get the long term private key for a new principal whose identity string is ID.
- **Send**( $\Pi_{i,j}^s, X$ ). This sends message  $X$  to the oracle  $\Pi_{i,j}^s$ . The output of  $\Pi_{i,j}^s$  is given to the adversary. The adversary can ask the principal  $\text{ID}_i$  to initiate a session with  $\text{ID}_j$  by a query **Send**( $\Pi_{i,j}^s, \lambda$ ) where  $\lambda$  is the empty string.
- **Reveal**( $\Pi_{i,j}^s$ ). This asks the oracle to reveal whatever session key it currently holds.
- **Corrupt**( $i$ ). This asks  $\text{ID}_i$  to reveal the long term private key  $d_{\text{ID}_i}$ .

The difference between the queries **Extract** and **Corrupt** is that the adversary can use **Extract** to get the private key for an identity string of her choice while **Corrupt** can only be used to get the private key of existing principals.

Let  $\Pi_{ij}^s$  be an initiator oracle (that is, it has received a  $\lambda$  message at the beginning) and  $\Pi_{ji}^{s'}$  be a responder oracle. If every message that  $\Pi_{ij}^s$  sends out is subsequently delivered to  $\Pi_{ji}^{s'}$ , with the response to this message being returned to  $\Pi_{ij}^s$  as the next message on its transcript, then we say the oracle  $\Pi_{ji}^{s'}$  matches  $\Pi_{ij}^s$ . Similarly, if every message that  $\Pi_{ji}^{s'}$  receives was previously generated by  $\Pi_{ij}^s$ , and each message that  $\Pi_{ji}^{s'}$  sends out is subsequently delivered to  $\Pi_{ij}^s$ , with the response to this message being returned to  $\Pi_{ji}^{s'}$  as the next message on its transcript, then we say the oracle  $\Pi_{ij}^s$  matches  $\Pi_{ji}^{s'}$ . The details for an exact definition of matching oracles could be found in [3].

For the definition of matching oracles, the reader should be aware the following scenarios: Even though the oracle  $\Pi_{ij}^s$  thinks that its matching oracle is  $\Pi_{ji}^{s'}$ , the real matching oracle for  $\Pi_{ij}^s$  could be  $\Pi_{ji}^{t'}$ . For example, if  $\Pi_{ij}^s$  sends a message  $X$  to  $\Pi_{ji}^{s'}$  and  $\Pi_{ji}^{s'}$  replies with  $Y$ . The adversary decides not to forward the message  $Y$  to  $\Pi_{ij}^s$ . Instead, the adversary sends the message  $X$  to initiate another oracle  $\Pi_{ji}^{t'}$  and  $\Pi_{ij}^s$  does not know the existence of this new oracle  $\Pi_{ji}^{t'}$ . The oracle  $\Pi_{ji}^{t'}$  replies with  $Y'$  and the adversary forwards this  $Y'$  to  $\Pi_{ij}^s$  as the responding message for  $X$ . In this case, the transcript of  $\Pi_{ij}^s$  matches the transcript of  $\Pi_{ji}^{t'}$ . Thus we consider  $\Pi_{ij}^s$  and  $\Pi_{ji}^{t'}$  as matching oracles. In another word, the matching oracles are mainly based the message transcripts.

In order to define the notion of a secure session key exchange, the adversary is given an additional experiment. That is, in addition to the above regular queries, the adversary can choose, at any time during its run, a **Test**( $\Pi_{i,j}^s$ ) query to a completed oracle  $\Pi_{i,j}^s$  with the following properties:

- The adversary has never issued, at any time during its run, the query **Extract**( $\text{ID}_i$ ) or **Extract**( $\text{ID}_j$ ).
- The adversary has never issued, at any time during its run, the query **Corrupt**( $i$ ) or **Corrupt**( $j$ ).
- The adversary has never issued, at any time during its run, the query **Reveal**( $\Pi_{i,j}^s$ ).
- The adversary has never issued, at any time during its run, the query **Reveal**( $\Pi_{j,i}^{s'}$ ) if the matching oracle  $\Pi_{j,i}^{s'}$  for  $\Pi_{i,j}^s$  exists (note that such an oracle may not exist if the adversary is impersonating the  $\text{ID}_j$  to the oracle  $\Pi_{i,j}^s$ ). The value of  $s$  may be different from the value of  $s'$  since the adversary may run fake sessions to impersonate any principals without victims' knowledge.

Let  $sk_{i,j}^s$  be the value of the session key held by the oracle  $\Pi_{i,j}^s$  that has been established between  $\text{ID}_i$  and  $\text{ID}_j$ . The oracle  $\Pi_{i,j}^s$  tosses a coin  $b \leftarrow_R \{0, 1\}$ . If  $b = 1$ , the adversary is given  $sk_{i,j}^s$ . Otherwise, the adversary is given a value  $r$  randomly chosen from the probability distribution of keys generated by the protocol. In the end, the attacker outputs a bit  $b'$ . The advantage that the adversary has for the above guess is defined as

$$\text{Adv}^A(k) = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

Now we are ready to give the exact definition for a secure key agreement protocol.

**Definition 2.** A key agreement protocol  $\Pi$  is *BR-secure* if the following conditions are satisfied for any adversary:

1. If two uncorrupted oracles  $\Pi_{ij}^s$  and  $\Pi_{ji}^{s'}$  have matching conversations (e.g., the adversary is passive) and both of them are complete according to the protocol  $\Pi$ , then both oracles will always accept and hold the same session key which is uniformly distributed over the key space.
2.  $\text{Adv}^{\mathcal{A}}(k)$  is negligible.

In the following, we briefly discuss the attributes that a BR-secure key agreement protocol achieves.

- **Known session keys.** The adversary may use **Reveal**( $\Pi_{ij}^{s'}$ ) query before or after the query **Test**( $\Pi_{ij}^s$ ). Thus in a secure key agreement model, the adversary learns zero information about a fresh key for session  $s$  even if she has learnt keys for other sessions  $s'$ .
- **Impersonation attack.** If the adversary impersonates  $\text{ID}_j$  to  $\text{ID}_i$ , then she still learns zero information about the session key that the oracle  $\Pi_{ij}^s$  holds for this impersonated  $\text{ID}_j$  since there is no matching oracle for  $\Pi_{ij}^s$  in this scenario. Thus  $\mathcal{A}$  can use **Test** query to test this session key that  $\Pi_{ij}^s$  holds.
- **Unknown key share.** If  $\text{ID}_i$  establishes a session key with  $\text{ID}_l$  though he believes that he is talking to  $\text{ID}_j$ , then there is an oracle  $\Pi_{ij}^s$  that holds this session key  $sk_{ij}$ . At the same time, there is an oracle  $\Pi_{li'}^{s'}$  that holds this session key  $sk_{ij}$ , for some  $i'$  (normally  $i' = i$ ). During an unknown key share attack, the user  $\text{ID}_j$  may not know this session key. Since  $\Pi_{ij}^s$  and  $\Pi_{li'}^{s'}$  are not matching oracles, the adversary can make the query **Reveal**( $\Pi_{li'}^{s'}$ ) to learn this session key before the query **Test**( $\Pi_{ij}^s$ ). Thus the adversary will succeed for this **Test** query challenge if the unknown key share attack is possible.

However, the following important security properties that a secure key agreement scheme should have are not implied from the original BR-security model.

- **Perfect forward secrecy.** This property requires that previously agreed session keys should remain secret, even if both parties' long-term private key materials are compromised. Bellare-Rogaway model does not capture this property. Canetti and Krawczyk's model [9] use the session-key expiration primitive to capture this property. Similar modification to Bellare-Rogaway model are required to capture this property also. We will give a separate proof that the IDAK key agreement protocol achieves weak perfect forward secrecy. Note that as pointed out in [19], no two-message key-exchange protocol authenticated with public keys and with no secure shared state can achieve perfect forward secrecy.
- **Key compromise impersonation resilience.** If the entity  $A$ 's long term private key is compromised, then the adversary could impersonate  $A$  to others, but it should not be able to impersonate others to  $A$ . Similar to wPFS property, Bellare-Rogaway model does not capture this property. We will give a separate proof that the IDAK key agreement protocol has this property.



## 5 The security of IDAK

Before we present the security proof for the IDAK key agreement protocol, we first prove some preliminary results that will be used in the security proof.

**Lemma 3.** *Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family,  $G_\rho = \langle G, G_1, \hat{e} \rangle$ ,  $g$  be a random generator of  $G$ , and  $\pi : G \times G \rightarrow Z_q$  be a random oracle. Assume DBDH-Assumption holds for  $\mathcal{G}$  and let  $\mathcal{X}$  and  $\mathcal{Y}$  be two distributions defined as*

$$\mathcal{X} = \langle \mathcal{R}, g^{\beta x_0}, g^{\gamma y_0}, \hat{e}(g, g)^{(x_0 + \pi(g^{\beta x_0}, g^{\gamma y_0}))(y_0 + \pi(g^{\gamma y_0}, g^{\beta x_0}))\alpha\beta\gamma}, \hat{e}(g, g)^{\alpha\beta\gamma} \rangle$$

and  $\mathcal{Y} = \langle \mathcal{R}, g^{\beta x_0}, g^{\gamma y_0}, \hat{e}(g, g)^{(x_0 + \pi(g^{\beta x_0}, g^{\gamma y_0}))(y_0 + \pi(g^{\gamma y_0}, g^{\beta x_0}))t}, \hat{e}(g, g)^t \rangle$

Then we have

1. The two distributions  $\mathcal{X}$  and  $\mathcal{Y}$  are computationally indistinguishable if  $\mathcal{R}$  is defined as

$$\mathcal{R} = \left( g, g^\alpha, g^\beta, g^\gamma, g^x, g^r, g_A, \hat{e} \left( g^{x + \beta\pi(g^x, g_A)}, g_A \cdot g^{r\pi(g_A, g^x)} \right)^\alpha \right),$$

$\alpha, \beta, \gamma, x, t, x_0$  are chosen from  $Z_q^*$  uniformly,  $g^r = g^\gamma$  or  $r$  is either chosen from  $Z_q^*$  uniformly,  $g_A$  and  $g^{\gamma y_0}$  are chosen from  $G$  within polynomial time according to a fixed distribution given the view  $(g^x, g^r, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0})$  without violating DBDH-Assumption.

2. For any constant  $m \leq \text{poly}(k)$ , the two distributions  $\mathcal{X}$  and  $\mathcal{Y}$  are computationally indistinguishable if  $\mathcal{R}$  is defined as:

$$(g, g^\alpha, g^\beta, g^\gamma, (g^{x_i}, g^{r_j}, g_{A,l})_{i,j,l \leq m}, (\hat{e}(g^{x_i + \beta\pi(g^{x_i}, g_{A,l})}, g_{A,l} \cdot g^{r_j\pi(g_{A,l}, g^{x_i})})^\alpha : i, j, l \leq m))$$

where  $\alpha, \beta, \gamma, x_i$  are uniformly chosen from  $Z_q^*$ ,  $r_j$  are either chosen from  $Z_q^*$  uniformly or  $g^{r_j} = g^\gamma$ , and  $g_{A,l}$  is chosen within polynomial time according to a fixed distribution given the view  $(g^{x_i}, g^{r_j}, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0} : i, j, l \leq m)$  without violating DBDH-Assumption.

3. For any constant  $m \leq \text{poly}(k)$ , the two distributions  $\mathcal{X}$  and  $\mathcal{Y}$  are computationally indistinguishable if  $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$ , where  $\mathcal{R}_1$  is defined as the  $\mathcal{R}$  in the item 2, and  $\mathcal{R}_2$  is defined as:

$$((g_{A,i}, g^{r_j}, g_{A,l})_{i,j,l \leq m}, (\hat{e}(g_{A,i} \cdot g^{\beta\pi(g_{A,i}, g_{A,l})}, g_{A,l} \cdot g^{r_j\pi(g_{A,l}, g_{A,i})})^\alpha : i, j, l \leq m))$$

where  $r_j$  are either chosen from  $Z_q^*$  uniformly or  $g^{r_j} = g^\gamma$ ,  $g_{A,i}$  and  $g_{A,l}$  are chosen within polynomial time according to a fixed distribution given the view  $(g^{x_i}, g^{r_j}, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0}, g^{\gamma y_0} : i, j, l \leq m)$  without violating DBDH-Assumption and with the condition that " $g_{A,i} \neq g^{\beta x_0}$  or  $g_{A,l} \neq g^{\gamma y_0}$ ". Note that  $g_{A,i}$  and  $g_{A,l}$  could have different distributions.

**Proof.** See Appendix. □

**Theorem 3.** *Suppose that the functions  $H$  and  $\pi$  are random oracles and the bilinear group family  $\mathcal{G}$  satisfies DBDH-Assumption. Then the IDAK scheme is a BR-secure key agreement protocol.*

**Proof.** See Appendix. □

## 6 Weak Perfect forward secrecy

In this section, we show that the protocol IDAK achieves weak perfect forward secrecy property. Perfect forward secrecy property requires that even if Alice and Bob lose their private keys  $d_{ID_A} = g_{ID_A}^\alpha$  and  $d_{ID_B} = g_{ID_B}^\alpha$ , the session keys established by Alice and Bob in the previous sessions are still secure. Krawczyk [19] pointed out that no two-message key-exchange protocol authenticated with public keys and with no secure shared state can achieve perfect forward secrecy. Weak perfect forward secrecy (wPFS) property for key agreement protocols sates as follows [19]: any session key established by uncorrupted parties without active intervention by the adversary is guaranteed to remain secure even if the parties to the exchange are corrupted after the session key was erased from the parties memory (for a formal definition, the reader is referred to [19]).

In the following, we show the IDAK achieves wPFS property. Using the similar primitive of “session-key expiration” as in Canetti and Krawczyk’s model [9], we can revise Bellare-Rogaway model so that wPFS property is provable also. In Bellare-Rogaway model, the  $\text{Test}(\Pi_{i,j}^s)$  query is allowed only if the four properties in Section 4 are satisfied. We can replace the property “the adversary has never issued, at any time during its run, the query  $\text{Corrupt}(i)$  or  $\text{Corrupt}(j)$ ” with the property “the adversary has never issued, before the session  $\Pi_{i,j}^s$  is complete, the query  $\text{Corrupt}(i)$  or  $\text{Corrupt}(j)$ ”. We call this model the wpfsBR model. In the final version of this paper, we will show that the protocol IDAK is secure in the wpfsBR model. Thus IDAK achieves wPFS property. In the following, we present the essential technique used in the proof. It is essentially sufficient to show that the two distributions  $(\mathcal{R}, \hat{e}(g_{ID_A}, g_{ID_B})^z)$  and  $(\mathcal{R}, \hat{e}(g_{ID_A}, g_{ID_B})^{(x+\pi(g_{ID_A}^x, g_{ID_B}^y))(y+\pi(g_{ID_B}^y, g_{ID_A}^x))^\alpha})$  are computationally indistinguishable for  $\mathcal{R} = (g_{ID_A}^\alpha, g_{ID_B}^\alpha, g_{ID_A}^x, g_{ID_B}^y)$  and uniform at random chosen  $g_{ID_A}, g_{ID_B}, x, y, z, \alpha$ . Consequently, it is sufficient to prove the following theorem.

**Theorem 4.** *Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family,  $G_\rho = \langle G, G_1, \hat{e} \rangle$ . Assume that DBDH-Assumption holds for  $\mathcal{G}$ . Then the two distributions*

$$\begin{aligned} \mathcal{X} &= (g_1, g_2, g_1^\alpha, g_2^\alpha, g_1^x, g_2^y, \hat{e}(g_1, g_2)^{xy\alpha}) \\ \text{and} \quad \mathcal{Y} &= (g_1, g_2, g_1^\alpha, g_2^\alpha, g_1^x, g_2^y, \hat{e}(g_1, g_2)^z) \end{aligned}$$

*are computationally indistinguishable for random chosen  $g_1, g_2, x, y, z, \alpha$ .*

**Proof.** We use a random reduction. For a contradiction, assume that there is a polynomial time probabilistic algorithm  $\mathcal{D}$  that distinguishes  $\mathcal{X}$  and  $\mathcal{Y}$  with a non-negligible probability  $\delta_k$ . We construct a polynomial time probabilistic algorithm  $\mathcal{A}$  that distinguishes  $(\mathcal{R}, \hat{e}(g, g)^t)$  and  $(\mathcal{R}, \hat{e}(g, g)^{uvw})$  with  $\delta_k$ , where  $\mathcal{R} = (g, g^u, g^v, g^w)$  and  $u, v, w, t$  are uniformly at random in  $Z_q$ . Let the input of  $\mathcal{A}$  be  $(\mathcal{R}, \hat{e}(g, g)^{\tilde{t}})$ , where  $\tilde{t}$  is either  $uvw$  or uniformly at random in  $Z_q$ . We construct  $\mathcal{A}$  as follows.  $\mathcal{A}$  chooses random  $c_1, c_2, c_3, c_4, c_5 \in Z_q$  and sets  $g_1 = g^{c_1}, g_2 = g^{c_2}, g_1^\alpha = g^{uc_1c_3}, g_2^\alpha = g^{uc_2c_3}, g_1^x = g^{vc_1c_4}, g_2^y = g^{wc_2c_5}$ , and  $\hat{e}(g_1, g_2)^{\tilde{z}} = \hat{e}(g, g)^{\tilde{t}c_1c_2c_3c_4c_5}$ . Let  $\mathcal{A}(\mathcal{R}, \hat{e}(g, g)^{\tilde{t}}) = \mathcal{D}(g_1, g_2, g_1^\alpha, g_2^\alpha, g_1^x, g_2^y, \hat{e}(g_1, g_2)^{\tilde{z}})$ . Note that if  $\tilde{t} = uvw$ , then  $c_1, c_2, \alpha, x, y$  are uniform in  $Z_q$  (and independent of each other and of  $u, v, w$ ) and  $xy\alpha = \tilde{z}$ . Otherwise,

$c_1, c_2, \alpha, x, y$  are uniform in  $Z_q$  and independent of each other and of  $u, v, w$ . Therefore, by the definitions,

$$\begin{aligned} \Pr[\mathcal{A}(\mathcal{R}, \hat{e}(g, g)^{uvw}) = 1] &= \Pr[\mathcal{D}(\mathcal{X}) = 1] \\ \text{and} \quad \Pr[\mathcal{A}(\mathcal{R}, \hat{e}(g, g)^t) = 1] &= \Pr[\mathcal{D}(\mathcal{Y}) = 1] \end{aligned}$$

Thus  $\mathcal{A}$  distinguishes  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^t \rangle$  and  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^{uvw} \rangle$  with  $\delta_k$ . This is a contradiction.  $\square$

Though Theorem 4 shows that the protocol IDAK achieves weak perfect forward secrecy even if both participating parties' long term private keys were corrupted, IDAK does not have perfect forward secrecy when the master secret  $\alpha$  were leaked. The perfect forward secrecy against the corruption of  $\alpha$  could be achieved by requiring Bob (the responder in the IDAK protocol) to send  $g_{ID_A}^y$  in addition to the value  $R_B = g_{ID_B}^y$  and by requiring both parties to compute the shared secret as  $H(g_{ID_A}^{xy} || sk_{AB})$  where  $sk_{AB}$  is the shared secret established by the IDAK protocol.

## 7 Key compromise impersonation (KCI) resilience

In this section, we informally show that the protocol IDAK has the key compromise impersonation resilience property. That is, if Alice loses her private key  $d_A = g_{ID_A}^\alpha$ , then the adversary still could not impersonate Bob to Alice. For a formal proof of KCI, we still need to consider the information obtained by the adversary by **Reveal**, **Extract**, **Send**, **Corrupt** queries in other sessions. This will be done in the final version of this paper.

In order to show KCI for IDAK, it is (informally) sufficient to show that the two distributions  $\left( \mathcal{R}, \hat{e} \left( g_{ID_A}^x \cdot g_{ID_A}^{\pi(g_{ID_A}^x, R_B)}, R_B \cdot g_{ID_B}^{\pi(R_B, g_{ID_A}^x)} \right)^\alpha \right)$  and  $(\mathcal{R}, \hat{e}(g_{ID_A}, g_{ID_B})^z)$  are computationally indistinguishable for  $\mathcal{R} = (g_{ID_A}^\alpha, g_{ID_A}^x, R_B)$ , where  $g_{ID_A}, g_{ID_B}, x, z, \alpha$  are chosen uniform at random, and  $R_B$  is chosen according to some probabilistic polynomial time distribution. Since the value  $\hat{e} \left( g_{ID_A}^{\pi(g_{ID_A}^x, R_B)}, R_B \cdot g_{ID_B}^{\pi(R_B, g_{ID_A}^x)} \right)^\alpha$  is known, it is sufficient to prove the following theorem.

**Theorem 5.** *Let  $\mathcal{G} = \{G_\rho\}$  be a bilinear group family,  $G_\rho = \langle G, G_1, \hat{e} \rangle$ . Assume that DBDH-Assumption holds for  $\mathcal{G}$ . Then the two distributions*

$$\begin{aligned} \mathcal{X} &= \left( g_1, g_2, g_1^\alpha, g_1^x, R_B, \hat{e} \left( g_1^x, R_B \cdot g_2^{\pi(R_B, g_1^x)} \right)^\alpha \right) \\ \text{and} \quad \mathcal{Y} &= (g_1, g_2, g_1^\alpha, g_1^x, R_B, \hat{e}(g_1, g_2)^z) \end{aligned}$$

*are computationally indistinguishable for random chosen  $g_1, g_2, x, z, \alpha$ , where  $R_B$  is chosen according to some probabilistic polynomial time distribution.*

**Proof.** Since  $g_1^x$  is chosen uniform at random, and  $\pi$  is a random oracle, we may assume that  $R_B \cdot g_2^{\pi(R_B, g_1^x)}$  is uniformly distributed over  $G$  when  $R_B$  is chosen according to any probabilistic polynomial time distribution. Thus the proof is similar to the proof of Theorem 4 and the details are omitted. The theorem could also be proved using the

Splitting lemma [28] which was used to prove the fork lemma. Briefly, the Splitting lemma translates the fact that when a subset  $A$  is “large” in a product space  $X \times Y$ , it has many large sections. Using the Splitting lemma, one can show that if  $\mathcal{D}$  can distinguish  $\mathcal{X}$  and  $\mathcal{Y}$ , then by replaying  $\mathcal{D}$  with different random oracle  $\pi$ , one can get sufficient many tuples  $(g_1, g_2, g_1^\alpha, g_1^x, R_B, \pi_1, \pi_2)$  such that (1)  $\pi_1(R_B, g_1^x) \neq \pi_2(R_B, g_1^x)$ ; (2)  $\mathcal{D}$  distinguishes  $\mathcal{X}_1$  and  $\mathcal{Y}$  (respectively  $\mathcal{X}_2$  and  $\mathcal{Y}$ ) when  $z$  is uniformly chosen but other values takes the values from the above tuple with  $\pi_1$  (respectively  $\pi_2$ ). Since  $\hat{e}\left(g_1^x, R_B \cdot g_2^{\pi_1(R_B, g_1^x)}\right)^\alpha / \hat{e}\left(g_1^x, R_B \cdot g_2^{\pi_2(R_B, g_1^x)}\right)^\alpha = \hat{e}(g_1, g_2)^{x\alpha(\pi_1(R_B, g_1^x) - \pi_2(R_B, g_1^x))}$ . Thus, for the above tuple, we can distinguish  $\hat{e}(g_1, g_2)^{x\alpha}$  from  $\hat{e}(g, g)^z$  for random chosen  $z$ . This is a contradiction with the DBDH-Assumption.  $\square$

## References

1. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In: *Advances in Cryptology, Crypto 96*, pages 1–15, 1996.
2. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In: *30th Annual ACM Symposium on Theory of Computing*, 1998.
3. M. Bellare and P. Rogaway. Random oracles are practical: a paradigms for designing efficient protocols. In: *Proc. 1st ACM Conference on Computer Communication Security*, pages 62–73, ACM Press, 1993.
4. M. Bellare and P. Rogaway. Entity authentication and key distribution. In: *Advances in Cryptology, Crypto 93*, LNCS 773 (1993), 232–249.
5. M. Blum and S. Micali. How to generate cryptographically strong sequence of pseudo-random bits. *SIAM J. Comput.* **13**:850–864, 1984.
6. D. Boneh. The decision Diffie-Hellman problem. In: *ANTS-III*, LNCS 1423 (1998), 48–63.
7. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing* **32**(3):586–615, 2003.
8. R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In: *Proc. 42nd FOCS*, 2001.
9. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In: *Advances in Cryptology, Eurocrypt 01*, LNCS 2045 (2001), 453–474. Full version available from Cryptology ePrint Archive 2001-040 (<http://eprint.iacr.org/>).
10. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In: *Eurocrypt 02*.
11. L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairing. In: *Proc. 16th IEEE Security Foundations Workshop*, pages 219–233. IEEE Computer Society Press, 2003.
12. Z. Cheng and L. Chen. On the security proof of McCullagh-Barreto’s key agreement protocol and its variants. <http://eprint.iacr.org/2005/201.pdf>
13. Z. Cheng, M. Nistazakis, R. Comley, and L. Vasiu. On indistinguishability-based security model of key agreement protocols-simple cases. In *Proc. of ACNS 04*, June 2004.
14. K. Choo. Revisit of McCullagh-Barreto two party ID-based authentication key agreement protocols. <http://eprint.iacr.org/2004/343.pdf>
15. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **6**(1976), 644–654.
16. A. Fiat and A. Shamir. How to prove yourself: practical solutions of identification and signature problems. In: *Advances in Cryptology, Crypto 86*, LNCS 263 (1987), 186–194.

17. M. Girault and J. Pailles. An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. In: *Proc. ESORICS 90*, pages 173–184.
18. A. Joux. A one round protocol for tripartite Diffie-Hellman. In: *Algorithmic number theory symposium, ANTS-IV*, LNCS 1838, pages 385–394, 2000.
19. H. Krawczyk. HMQV: a high-performance secure Diffie-Hellman protocol. In: *Proc. Crypto 05*, Springer, 2005.
20. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, **28**(2):119–134.
21. S. Li, Q. Yuan, and J. Li. Towards security two-part authenticated key agreement protocols. <http://eprint.iacr.org/2005/300.pdf>.
22. P. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. *Proc. of CT-RSA 2005*, pages 262–274, LNCS 3376, Springer Verlag, 2005.
23. P. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. <http://eprint.iacr.org/2004/122.pdf>
24. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In: *38th Annual Symposium on Foundations of Computer Science*, IEEE Press, 1998.
25. V. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, **55**(1994), 165–172.
26. NIST Special Publication 800-56: Recommendation on key establishment schemes, draft 2.0, 2003. <http://csrc.nist.gov/CryptoToolkit/kms/key schemes-Jan03.pdf>.
27. E. Okamoto. Proposal for identity-based key distribution system. *Electronics Letters* **22**:1283–1284, 1986.
28. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology* **13**(3):361–396, 2000.
29. E. Ryu, E. Yoon, and K. Yoo. An efficient ID-based authenticated key agreement protocol from pairing. In: *Networking 2004*, pages 1458–1463, LNCS 3042, Springer Verlag, 2004.
30. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In: *2000 Symp. on Cryptography and Information Security (SCIS 2000)*, Okinawa, Japan 2000.
31. M. Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. <http://eprint.iacr.org/2002/164.pdf>
32. A. Shamir. Identity-based cryptosystems and signature schemes. In: *Advances in Cryptology, Crypto 84*, LNCS 196, pages 47–53, Springer Verlag 1984.
33. K. Shim. Efficient ID-based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters* **39**(8):653–654, 2003.
34. V. Shoup. Lower bounds for discrete logarithms and related problems. In: *Advances in Cryptology, Eurocrypt 97*, LNCS 1233 (1997), 256–266.
35. V. Shoup. On formal models for secure key exchange. IBM Technical Report RZ 3120, 1999.
36. N. P. Smart. Identity-based authenticated key agreement protocol based on Weil pairing. *Electronics Letters* **38**(13):630–632, 2002.
37. S. Sun and B. Hsieh. Security analysis of Shim’s authenticated key agreement protocols from pairing. <http://eprint.iacr.org/2003/113.pdf>
38. K. Tanaka and E. Okamoto. Key distribution system for mail systems using ID-related information directory. *Computers and Security* **10**:25–33, 1991.
39. G. Xie. Cryptanalysis of Noel McCullagh and Paulo S. L. M. Barreto’s two-party identity-based key agreement. <http://eprint.iacr.org/2004/308.pdf>
40. G. Xie. An ID-based key agreement scheme from pairing. <http://eprint.iacr.org/2005/093.pdf>

## 8 Appendix

### 8.1 Proof of Theorem 1

The fact that the CBDH-Assumption implies the perfect-CBDH-Assumption is trivial. The converse is proved by the self-random-reduction technique (see [5,24]). Let  $\mathcal{O}$  be a CBDH oracle. That is, there exists a  $c > 0$  such that (1) holds with  $\mathcal{C}$  replaced with  $\mathcal{O}$ . We construct a perfect-CBDH algorithm  $\mathcal{C}$  which makes use of the oracle  $\mathcal{O}$ . Given  $g, g^x, g^y, g^z \in G$ , algorithm  $\mathcal{C}$  must compute  $\hat{e}(g, g)^{xyz}$  with overwhelming probability. Consider the following algorithm: select  $a, b, c \in_R Z_q$  (unless stated explicitly, we use  $x \in_R X$  to denote that  $x$  is randomly chosen from  $X$  in the remainder of this paper) and output

$$I_{x,y,z,a,b,c} = \mathcal{O}(g, g^{x+a}, g^{y+b}, g^{z+c}) \cdot \hat{e}(g, g)^{-(abz+abc+ayz+ayc+xbz+xbc+xyz)}.$$

One can easily verify that if  $\mathcal{O}(g, g^{x+a}, g^{y+b}, g^{z+c}) = \hat{e}(g, g)^{(x+a)(y+b)(z+c)}$ , then  $I_{x,y,z,a,b,c} = \hat{e}(g, g)^{xyz}$ . Consequently, standard amplification techniques can be used to construct the algorithm  $\mathcal{C}$ . The details are omitted.

### 8.2 Proof of Lemma 2

For a contradiction, assume that there is a probabilistic polynomial-time algorithm  $\mathcal{D}$  that distinguishes the two distributions  $\mathcal{X}_2$  and  $\mathcal{Y}_2$  with non-negligible probability  $\delta_k$ . In the following we construct a probabilistic polynomial-time algorithm  $\mathcal{D}'$  to distinguish the two distributions  $\mathcal{X}_1$  and  $\mathcal{Y}_1$ .  $\mathcal{D}'$  is defined by letting  $\mathcal{D}'(\mathcal{R}, X, Y) = \mathcal{D}(\mathcal{R}, X \cdot Y)$  for all  $\mathcal{R}$ , and  $X, Y \in G_1$ . By this definition, we have  $\Pr[\mathcal{D}'_r(\mathcal{X}_1) = 1 | \mathcal{R}, r] = \Pr[\mathcal{D}_r(\mathcal{X}_2) = 1 | \mathcal{R}, r]$ , for any fixed internal coin tosses  $r$  of  $\mathcal{D}$  and  $\mathcal{D}'$ .

Let  $D_{\mathcal{R},r}^{\mathcal{D}} = \{X : \mathcal{D}_r(\mathcal{R}, X) = 1\}$  and  $D_{\mathcal{R},r}^{\mathcal{D}'} = \{(X, Y) : \mathcal{D}'_r(\mathcal{R}, X, Y) = 1\}$ . By definition of  $\mathcal{D}'$ , we have  $D_{\mathcal{R},r}^{\mathcal{D}'} = \{(X, Y) : X \cdot Y \in D_{\mathcal{R},r}^{\mathcal{D}}\}$ . It follows that  $|D_{\mathcal{R},r}^{\mathcal{D}'}| = q|D_{\mathcal{R},r}^{\mathcal{D}}|$  and  $\Pr[\mathcal{D}'_r(\mathcal{Y}_1) = 1 | \mathcal{R}, r] = |D_{\mathcal{R},r}^{\mathcal{D}'}|/q^2 = |D_{\mathcal{R},r}^{\mathcal{D}}|/q = \Pr[\mathcal{D}_r(\mathcal{Y}_2) = 1 | \mathcal{R}, r]$ . Thus we have

$$\begin{aligned} & |\Pr[\mathcal{D}'(\mathcal{X}_1) = 1] - \Pr[\mathcal{D}'(\mathcal{Y}_1) = 1]| \\ &= \left| \sum_{\mathcal{R}, r} \Pr[\mathcal{R}, r] \cdot (\Pr[\mathcal{D}'_r(\mathcal{X}_1) = 1 | \mathcal{R}, r] - \Pr[\mathcal{D}'_r(\mathcal{Y}_1) = 1 | \mathcal{R}, r]) \right| \\ &= \left| \sum_{\mathcal{R}, r} \Pr[\mathcal{R}, r] \cdot (\Pr[\mathcal{D}_r(\mathcal{X}_2) = 1 | \mathcal{R}, r] - \Pr[\mathcal{D}_r(\mathcal{Y}_2) = 1 | \mathcal{R}, r]) \right| \\ &= |\Pr[\mathcal{D}(\mathcal{X}_2) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_2) = 1]| \\ &> \delta_k. \end{aligned}$$

Hence,  $\mathcal{D}'$  distinguishes the distributions  $\mathcal{X}_1$  and  $\mathcal{Y}_1$  with non-negligible probability  $\delta_k$ . This contradicts the assumption of the Lemma.

### 8.3 Proof of Lemma 3

The Lemma could be proved using complicated version of the Splitting lemma by Pointcheval-Stern [28] (see the proof of Theorem 7). In the following, we use the random reduction to prove the lemma.

1. For a contradiction, assume that there is a polynomial time probabilistic algorithm  $\mathcal{D}$  that distinguishes  $\mathcal{X}$  and  $\mathcal{Y}$ . We construct a polynomial time probabilistic algorithm  $\mathcal{A}$  that distinguishes  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^a \rangle$  and  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^{uvw} \rangle$  with  $\delta_k$ , where  $u, v, w, a$  are uniformly at random in  $Z_q$ .

Let the input of  $\mathcal{A}$  be  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^{\tilde{a}} \rangle$ , where  $\tilde{a}$  is either  $uvw$  or uniformly at random in  $Z_q$ .  $\mathcal{A}$  chooses uniformly at random  $c_1, c_2, c_3, x, x_0 \in Z_q$ , sets  $g^\alpha = g^{c_1 u + c_2}$ ,  $g^\beta = g^{v + c_3}$ ,  $g^\gamma = g^{w + c_4}$ , chooses uniformly at random  $r \in Z_q$  or lets  $g^r = g^\beta$ , chooses  $g^{\gamma y_0}, g_{\mathcal{A}} \in G$  within polynomial time according to any distribution given the view  $(g^x, g^r, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0})$  (the distributions for  $g_{\mathcal{A}} \in G$  and  $g^{\gamma y_0}$  could be different). Since  $g^x$  and  $g^{\beta x_0}$  are uniformly chosen from  $G$ , we may assume that the values of  $\pi(g^x, g_{\mathcal{A}})$  and  $\pi(g^{\gamma y_0}, g^{\beta x_0})$  are unknown yet. Without loss of generality, we may assume that  $x + \beta\pi(g^x, g_{\mathcal{A}})$  and  $y_0 + \pi(g^{\gamma y_0}, g^{\beta x_0})$  take values  $c_5$  and  $c_6$  respectively, where  $c_5$  and  $c_6$  are uniformly chosen from  $Z_q$ . In a summary, the value of  $\mathcal{R}$  could be computed from  $g^u, g^v, g^w, c_1, c_2, c_3, c_4, c_5$  efficiently.  $\mathcal{A}$  then sets

$$\hat{e}(g, g)^{\tilde{t}} = \hat{e}(g, g)^{c_1 \tilde{a} + c_4(c_1 u + c_2)(v + c_3) + w(c_1 u c_3 + c_1 v + c_2 c_3)}.$$

$\mathcal{A}$  can compute  $\hat{e}(g, g)^{(x_0 + \pi(g^{\beta x_0}, g^{\gamma y_0}))(y_0 + \pi(g^{\gamma y_0}, g^{\beta x_0}))\tilde{t}}$  using the values of  $\hat{e}(g, g)^{\tilde{t}}$ ,  $x_0, \pi(g^{\beta x_0}, g^{\gamma y_0})$ ,  $c_6$ . Let  $\mathcal{A}(g, g^u, g^v, g^w, \hat{e}(g, g)^{\tilde{a}}) = \mathcal{D}(\mathcal{X})$ , where  $\mathcal{X}$  is obtained from  $\mathcal{Y}$  by replacing  $t$  with  $\tilde{t}$  and taking the remaining values as defined above.

Note that if  $\tilde{a} = uvw$ , then  $\tilde{t} = \alpha\beta\gamma$ , and  $\mathcal{X}$  is distributed according to the distribution  $\mathcal{X}$ . That is,  $\alpha, \beta, \gamma, x, x_0$  are uniform in  $Z_q$  and independent of each other and of  $(u, v, w)$ ,  $(r, g_{\mathcal{A}}, g^{\gamma y_0})$  is chosen according to the specified distributions without violating DBDH-Assumption. Otherwise,  $\mathcal{X}$  is distributed according to the distribution  $\mathcal{Y}$ , and  $\tilde{t}$  is uniform in  $Z_q$  and independent of  $\alpha, \beta, \gamma, x, x_0, r, u, v, w, g_{\mathcal{A}}, g^{\gamma y_0}$ . Therefore, by definitions,

$$\begin{aligned} \Pr[\mathcal{A}(g, g^u, g^v, g^w, \hat{e}(g, g)^{uvw}) = 1] &= \Pr[\mathcal{D}(\mathcal{X}) = 1] \\ \text{and} \quad \Pr[\mathcal{A}(g, g^u, g^v, g^w, \hat{e}(g, g)^a) = 1] &= \Pr[\mathcal{D}(\mathcal{Y}) = 1] \end{aligned}$$

Thus  $\mathcal{A}$  distinguishes  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^a \rangle$  and  $\langle g, g^u, g^v, g^w, \hat{e}(g, g)^{uvw} \rangle$  with  $\delta_k$ , where  $a$  is uniform at random in  $Z_q$ . This is a contradiction.

2. This part of the Lemma could be proved in the same way. The details are omitted.

3. Since " $g_{\mathcal{A}, i} \neq g^{\beta x_0}$  or  $g_{\mathcal{A}, l} \neq g^{\gamma y_0}$ ", we may assume that the values of  $\pi(g_{\mathcal{A}, i}, g_{\mathcal{A}, l})$  and  $\pi(g_{\mathcal{A}, l}, g_{\mathcal{A}, i})$  are unknown yet. By the random oracle property of  $\pi$ , this part of the Lemma could be proved in the same way as in item 1. The details are omitted.

## 9 Proof of Theorem 3

**Proof.** By Theorem 2, the condition 1 in the Definition 2 is satisfied for the IDAK key agreement protocol. In the following, we show that the condition 2 is also satisfied.

For a contradiction, assume that the adversary  $\mathcal{A}$  has non-negligible advantage  $\delta_k = \text{Adv}^{\mathcal{A}}(k)$  in guessing the value of  $b$  after the **Test** query. We show how to construct a simulator  $\mathcal{S}$  that uses  $\mathcal{A}$  as an oracle to distinguish the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  in the item 3 of Lemma 3 with non-negligible advantage  $2\delta_k(q_E - 2)^2/q_E^4$ , where  $q_E$  denotes the number of distinct **H-queries** that the algorithm  $\mathcal{A}$  has made.

The game between the challenger and the simulator  $\mathcal{S}$  starts with the challenger first generating bilinear groups  $G_\rho = \langle G, G_1, \hat{e} \rangle$  by running the algorithm **Instance Generator**. The challenger then chooses  $\alpha, \beta, \gamma, t \in_R Z_q$  and  $b \in_R \{0, 1\}$ . The challenger gives the tuple  $\langle \rho, g, g^\alpha, g^\beta, g^\gamma, \hat{e}(g, g)^{\tilde{t}} \rangle$  to the algorithm  $\mathcal{S}$  where  $\tilde{t} = \alpha\beta\gamma$  if  $b = 1$  and  $\tilde{t} = t$  otherwise. During the simulation, the algorithm  $\mathcal{S}$  can ask the challenger to provide randomly chosen  $g^{x_i}$ .  $\mathcal{S}$  may then choose (with the help of  $\mathcal{A}$  perhaps)  $g_{\mathcal{A},l}$  within polynomial time according to any distribution given the view  $(g^{x_i}, g^{r_j}, g^\alpha, g^\beta, g^\gamma, g^{\alpha x_0} : i, j, l \leq m)$  and sends  $g_{\mathcal{A},l}$  to the challenger. The challenger responds with  $\hat{e}(g^{x_i + \beta\pi(g^{x_i}, g_{\mathcal{A},l})}, g_{\mathcal{A},l} \cdot g^{r_j\pi(g_{\mathcal{A},l}, g^{x_i})})^\alpha$ . At the end of the simulation, the algorithm  $\mathcal{S}$  is supposed to output its guess  $b' \in \{0, 1\}$  for  $b$ . It should be noted that if  $b = 1$ , then the output of the challenger together with the values  $g_{\mathcal{A},l}$  selected by the simulator  $\mathcal{S}$  is the tuple  $\mathcal{X}$  of Lemma 3, and is the tuple  $\mathcal{Y}$  of Lemma 3 if  $b = 0$ . Thus the simulator  $\mathcal{S}$  could be used to distinguish  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3.

The algorithm  $\mathcal{S}$  selects two integers  $I, J \leq q_E$  randomly and works by interacting with  $\mathcal{A}$  as follows:

**Setup:** Algorithm  $\mathcal{S}$  gives  $\mathcal{A}$  the IDAK system parameters  $\langle q, G, G_1, \hat{e}, H, \pi \rangle$  where  $q, G, G_1, \hat{e}$  are parameters from the challenger,  $H$  and  $\pi$  are random oracles controlled by  $\mathcal{S}$  as follows.

**$H$ -queries:** At any time algorithm  $\mathcal{A}$  can query the random oracle  $H$  using the queries **Extract**( $ID_i$ ) or **GetID**( $ID_i$ ) =  $H(ID_i)$ . To respond to these queries algorithm  $\mathcal{S}$  maintains an  $H^{list}$  that contains a list of tuples  $\langle ID_i, g_{ID_i} \rangle$ . The list is initially empty. When  $\mathcal{A}$  queries the oracle  $H$  at a point  $ID_i$ ,  $\mathcal{S}$  responds as follows:

1. If the query  $ID_i$  appears on the  $H^{list}$  in a tuple  $\langle ID_i, g_{ID_i} \rangle$ , then  $\mathcal{S}$  responds with  $H(ID_i) = g_{ID_i}$ .
2. Otherwise, if this is the  $I$ -th new query of the random oracle  $H$ ,  $\mathcal{S}$  responds with  $g_{ID_i} = H(ID_i) = g^\beta$ , and adds the tuple  $\langle ID_i, g^\beta \rangle$  to the  $H^{list}$ . If this is the  $J$ -th new query of the random oracle,  $\mathcal{S}$  responds with  $g_{ID_i} = H(ID_i) = g^\gamma$ , and adds the tuple  $\langle ID_i, g^\gamma \rangle$  to the  $H^{list}$ .
3. In the remaining case,  $\mathcal{S}$  selects a random  $r_i \in Z_q$ , responds with  $g_{ID_i} = H(ID_i) = g^{r_i}$ , and adds the tuple  $\langle ID_i, g^{r_i} \rangle$  to the  $H^{list}$ .

**$\pi$ -queries:** At any time the challenger, the algorithm  $\mathcal{A}$ , and the algorithm  $\mathcal{S}$  can query the random oracle  $\pi$ . To respond to these queries algorithm  $\mathcal{S}$  maintains a  $\pi^{list}$  that contains a list of tuples  $\langle g_1, g_2, \pi(g_1, g_2) \rangle$ . The list is initially empty. When  $\mathcal{A}$  queries the oracle  $\pi$  at a point  $(g_1, g_2)$ ,  $\mathcal{S}$  responds as follows: If the query  $(g_1, g_2)$  appears on the  $\pi^{list}$  in a tuple  $\langle (g_1, g_2), \pi(g_1, g_2) \rangle$ , then  $\mathcal{S}$  responds with  $\pi(g_1, g_2)$ . Otherwise,  $\mathcal{S}$  selects a random  $v_i \in Z_q$ , responds with  $\pi(g_1, g_2) = v_i$ , and adds the tuple  $\langle (g_1, g_2), v_i \rangle$  to the  $\pi^{list}$ . Technically, the random oracle  $\pi$  could be held by an independent third party to avoid the confusion that the challenger also needs to access this random oracle also.

**Query phase:**  $\mathcal{S}$  responds to  $\mathcal{A}$ 's queries as follows.

For a **GetID**( $ID_i$ ) query,  $\mathcal{S}$  runs the  **$H$ -queries** to obtain a  $g_{ID_i}$  such that  $H(ID_i) = g_{ID_i}$ , and responds with  $g_{ID_i}$ .



For an **Extract**( $ID_i$ ) query for the long term private key, if  $i = I$  or  $i = J$ , then  $\mathcal{S}$  reports failure and terminates. Otherwise,  $\mathcal{S}$  runs the **H-queries** to obtain  $g_{ID_i} = H(ID_i) = g^{r_i}$ , and responds  $d_{ID_i} = (g^\alpha)^{r_i} = g_{ID_i}^\alpha$ .

For a **Send**( $\Pi_{i,j}^s, X$ ) query, we distinguish the following three cases:

1.  $X = \lambda$ . If  $i = I$  or  $J$ ,  $\mathcal{S}$  asks the challenger for a random  $R_i \in G$  (note that  $\mathcal{S}$  does not know the discrete logarithm of  $R_i$  with base  $g_{ID_i}$ ), otherwise  $\mathcal{S}$  chooses a random  $u_i \in Z_q^*$  and sets  $R_i = g_{ID_i}^{u_i}$ .  $\mathcal{S}$  lets  $\Pi_{i,j}^s$  reply with  $R_i$ . That is, we assume that  $ID_i$  is carrying out an IDAK key agreement protocol with  $ID_j$  and  $ID_i$  sends the first message  $R_i$  to  $ID_j$ .
2.  $X \neq \lambda$  and the transcript of the oracle  $\Pi_{i,j}^s$  is empty. In this case,  $\Pi_{i,j}^s$  is the responder to the protocol and has not sent out any message yet. If  $i = I$  or  $J$ ,  $\mathcal{S}$  asks the challenger for a random  $R_i \in G$ , otherwise  $\mathcal{S}$  chooses a random  $u_i \in Z_q^*$  and sets  $R_i = g_{ID_i}^{u_i}$ .  $\mathcal{S}$  lets  $\Pi_{i,j}^s$  reply with  $R_i$  and marks the oracle  $\Pi_{i,j}^s$  as completed.
3.  $X \neq \lambda$  and the transcript of the oracle  $\Pi_{i,j}^s$  is not empty. In this case,  $\Pi_{i,j}^s$  is the protocol initiator and should have sent out the first message already. Thus  $\Pi_{i,j}^s$  does not need to respond anything. After processing the query **Send**( $\Pi_{i,j}^s, X$ ),  $\mathcal{S}$  marks the oracle  $\Pi_{i,j}^s$  as completed.

For a **Reveal**( $\Pi_{i,j}^s$ ) query, if  $i \neq I$  and  $i \neq J$ ,  $\mathcal{S}$  computes the session key  $sk_{ij} = \hat{e}(g_{ID_j}^{\pi(R_j, R_i)} \cdot R_j, d_{ID_i}^{(u_i + \pi(R_i, R_j))})$  and responds with  $sk_{ij}$ , here  $R_j$  is the message received by  $\Pi_{i,j}^s$ . Note that the message  $R_j$  may not necessarily be sent by the oracle  $\Pi_{j,i}^{s'}$  for some  $s'$  since it could have been a bogus message from  $\mathcal{A}$ . Otherwise,  $i = I$  or  $i = J$ . Without loss of generality, we assume that  $i = I$ . In this case, the oracle  $\Pi_{I,j}^s$  does not know its private key  $g^{\beta\alpha}$ . Thus it needs help from the challenger to compute the shared session key. Let  $R_I$  and  $R_j$  be the messages that  $\Pi_{I,j}^s$  has sent out and received respectively.  $\Pi_{I,j}^s$  gives these two values to the challenger and the challenger computes the shared session key  $sk_{Ij} = \hat{e}\left(g_{ID_j}^{\pi(R_j, R_i)} \cdot R_j, R_I^{\alpha h} g^{\pi(R_I, R_j)\alpha\beta}\right)$ .  $\Pi_{I,j}^s$  then responds with  $sk_{Ij}$ .

For a **Corrupt**( $i$ ) query, if  $i = I$  or  $i = J$ , then  $\mathcal{S}$  reports failure and terminates. Otherwise,  $\mathcal{S}$  responds with  $d_{ID_i} = (g^\alpha)^{r_i} = g_{ID_i}^\alpha$ .

For the **Test**( $\Pi_{i,j}^s$ ) query, if  $i \neq I$  or  $j \neq J$ , then  $\mathcal{S}$  reports failure and terminates. Otherwise, assume that  $i = I$  and  $j = J$ . Let  $R_I = g_{ID_I}^{u_I}$  be the message that  $\Pi_{i,j}^s$  sends out (note that the challenger generated this message) and  $R_J = g_{ID_J}^{u_J}$  be the message that  $\Pi_{i,j}^s$  receives (note that  $R_J$  could be the message that the challenger generated or could be generated by the algorithm  $\mathcal{A}$ ).  $\mathcal{S}$  gives the messages  $R_I$  and  $R_J$  to the challenger. The challenger computes  $X = \hat{e}(g, g)^{(u_I + \pi(R_I, R_J))(u_J + \pi(R_J, R_I))\tilde{t}}$  and gives  $X$  to  $\mathcal{S}$ .  $\mathcal{S}$  responds with  $X$ . Note that if  $\tilde{t} = \alpha\beta\gamma$ , then  $X$  is the session key. Otherwise,  $X$  is a uniformly distributed group element.

**Guess:** After the **Test**( $\Pi_{i,j}^s$ ) query, the algorithm  $\mathcal{A}$  may issue other queries before finally outputs its guess  $b' \in \{0, 1\}$ . Algorithm  $\mathcal{S}$  outputs  $b'$  as its guess to the challenger.

**Claim:** If  $\mathcal{S}$  does not abort during the simulation then  $\mathcal{A}$ 's view is identical to its view in the real attack. Furthermore, if  $\mathcal{S}$  does not abort, then  $|\Pr[b = b'] - \frac{1}{2}| > \delta_k$ , where the probability is over all random coins used by  $\mathcal{S}$  and  $\mathcal{A}$ .

*Proof of Claim:* The responses to **H-queries** and  **$\pi$ -queries** are the same as in the real attack since the response is uniformly distributed. All responses to the **getID** queries, private key extract queries, message delivery queries, reveal queries, and corrupt queries are valid. It remains to show that the response to the test query is valid also. When  $\tilde{t}$  is uniformly distributed over  $Z_q$ , then Theorem 2 shows that  $X = \hat{e}(g, g)^{(u_I + \pi(R_I, R_J))(u_J + \pi(R_J, R_I))\tilde{t}}$  is uniformly distributed over  $G$  and is computationally indistinguishable from a random value before  $\mathcal{A}$ 's view. Therefore, by definition of the algorithm  $\mathcal{A}$ , we have  $|\Pr[b = b'] - \frac{1}{2}| > \delta_k$ .  $\square$

Suppose  $\mathcal{A}$  makes a total of  $q_E$  **H-queries**. We next calculate the probability that  $\mathcal{S}$  does not abort during the simulation. The probability that  $\mathcal{S}$  does not abort for **Extract** queries is  $(q_E - 2)/q_E$ . The probability that  $\mathcal{S}$  does not abort for **Corrupt** queries is  $(q_E - 2)/q_E$ . The probability that  $\mathcal{S}$  does not abort for **Test** queries is  $2/q_E^2$ . Therefore, the probability that  $\mathcal{S}$  does not abort during the simulation is  $2(q_E - 2)^2/q_E^4$ . This shows that  $\mathcal{S}$ 's advantage in distinguishing the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  in Lemma 3 is at least  $2\delta_k(q_E - 2)^2/q_E^4$  which is non-negligible.

To complete the proof of Theorem 3, it remains to show that the communications between  $\mathcal{S}$  and the challenger are carried out according to the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3. For a **Reveal**( $\Pi_{I,j}^s$ ) query, the challenger outputs  $\hat{e}\left(g_{\text{ID}_j}^{\pi(R_j, R_I)} \cdot R_j, R_I^{\alpha h} g^{\pi(R_I, R_j)\alpha\beta}\right)$  to the algorithm  $\mathcal{S}$ . Let  $R_I = g^x$ ,  $R_j = g_A$ , and  $g_{\text{ID}_j} = g^r$ . Then  $x$  is chosen uniform at random from  $Z_q$ ,  $r$  is chosen uniform at random from  $Z_q^*$  when  $j \neq J$  or  $r = \gamma$  when  $j = J$ , and the value of  $g_A$  is chosen by the algorithm  $\mathcal{A}$  or by the algorithm  $\mathcal{S}$  or by the challenger in probabilistic polynomial time according to the current views. For example, if  $g_A$  is chosen by the algorithm  $\mathcal{A}$ , then  $\mathcal{A}$  may generate  $g_A$  as the combination (e.g., multiplication) of some previously observed messages/values or generate it randomly. Thus the communication between the challenger and the algorithm  $\mathcal{S}$  during **Reveal**( $\Pi_{I,j}^s$ ) queries is carried out according to the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3. The case for **Reveal**( $\Pi_{J,j}^s$ ) queries is the same.

For the **Test**( $\Pi_{I,J}^s$ ) query, the challenger outputs  $X = \hat{e}(g, g)^{(u_I + \pi(R_I, R_J))(u_J + \pi(R_J, R_I))\tilde{t}}$  to the algorithm  $\mathcal{S}$ , where  $R_I = g^{\beta u_I}$  and  $R_J = g^{\gamma u_J}$ . Let  $x_0 = u_I$  and  $y_0 = u_J$ . Then  $x_0$  is chosen uniform at random from  $Z_q$  and the value of  $g^{\gamma y_0}$  is chosen by the algorithm  $\mathcal{A}$  or by the challenger in probabilistic polynomial time according to the current views. Similarly,  $\mathcal{A}$  may choose  $g^{\gamma y_0}$  as the combination (e.g., multiplication) of some previously observed messages/values. The communication between the challenger and the algorithm  $\mathcal{S}$  during the **Test**( $\Pi_{I,J}^s$ ) query is carried out according to the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3.

It should be noted that after the **Test**( $\Pi_{I,J}^s$ ) query, the adversary may create bogus oracles for the participants  $\text{ID}_I$  and  $\text{ID}_J$  and send bogus messages that may depend on all existing communicated messages (including messages held by the oracle  $\Pi_{I,J}^s$ ) and then reveal session keys from these oracles. In particular, the adversary may play a man in the middle attack by modifying the messages sent from  $\Pi_{I,J}^s$  to  $\Pi_{J,I}^{s'}$  and modifying the messages sent from  $\Pi_{J,I}^{s'}$  to  $\Pi_{I,J}^s$ . Then the oracles  $\Pi_{J,I}^{s'}$  and  $\Pi_{I,J}^s$  are not matching oracles. Thus  $\mathcal{A}$  can reveal the session key held by the oracle  $\Pi_{J,I}^{s'}$  before the guess. In the  $\mathcal{R}_2$  part in the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3, we have the condition " $g_{A,i} \neq g^{\beta x_0}$  or  $g_{A,l} \neq g^{\gamma y_0}$ " (this condition holds since the algorithm  $\mathcal{A}$  has not

revealed the matching oracles for  $\Pi_{I,J}^s$ ). If both  $g_{\mathcal{A},i} \neq g^{\beta x_0}$  and  $g_{\mathcal{A},l} \neq g^{\gamma y_0}$ , then the oracle  $\Pi_{J,I}^{s'}$  is a matching oracle for  $\Pi_{I,J}^s$  and  $\mathcal{A}$  is not allowed to reveal the session key held by the oracle  $\Pi_{J,I}^{s'}$ . Thus the communication between the challenger and the algorithm  $\mathcal{S}$  during these  $\text{Test}(\Pi_{I,J}^s)$  query is carried out according to the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3.

In the summary, all communications between the challenger and  $\mathcal{S}$  are carried out according to the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  of Lemma 3. This completes the proof of the Theorem.  $\square$

## 10 Practical considerations and applications

### 10.1 The function $\pi$

Though in the security proof of IDAK key agreement protocol,  $\pi$  is considered as a random oracle. In practice, we can use following simplified  $\pi$  functions.

- $\pi$  is a random oracle (secure hash function) from  $G \times G$  to  $Z_{2^{\lceil \log q \rceil / c}}^*$  (e.g.,  $c = 2$ ).
- If  $g_1 = (x_{g_1}, y_{g_1}), g_2 = (x_{g_2}, y_{g_2}) \in G$  are points on an elliptic curve, then let  $\pi(g_1, g_2) = \bar{x}_g \bmod 2^{\lceil x_g \rceil / 2}$  where  $\bar{x}_g = x_{g_1} \oplus x_{g_2}$ . That is,  $\pi(g_1, g_2)$  is the exclusive-or of the second half parts of the first coordinates of the elliptic curve points  $g_1$  and  $g_2$ .
- $\pi$  is a random oracle that the output only depends on the the first input variable or any of the above function restricted in such a way that the output only depends on the the first input variable. In another word,  $\pi : G \rightarrow Z_q^*$ .

It should be noted any  $\pi$  function, for which Lemma 3 holds, can be used in the IDAK protocol. Though we do not know whether Lemma 3 holds for  $\pi$  functions that we have listed above, we have strong evidence that this is true. First, if we assume that the group  $G_2$  is a generic group in the sense of Nechaev [25] and Shoup [34]. Then we can prove that Lemma 3 holds for the above  $\pi$  functions. Secondly, if the distribution  $\mathcal{G}(g^x, g^r, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0})$  in Lemma 3 is restricted to the distribution:

$$\{g^{f(x,r,\alpha,\beta,\gamma,\beta x_0,\mathbf{y})} : f \text{ is a linear function, } \mathbf{y} \text{ is a tuple of uniformly random values from } Z_q\}.$$

Then we can prove that Lemma 3 holds for the above  $\pi$  functions. We may conjecture that the adversary algorithm  $\mathcal{A}$  can only generate  $g_{\mathcal{A}}$  and  $g^{\gamma y_0}$  according to the above distribution unless CDH-Assumption fails for  $G$ . Thus, under this conjecture (without the condition that  $G_2$  is a generic group), the above list of  $\pi$  functions can be used in IDAK protocol securely.

### 10.2 Performance

Our analysis in this section will be based on the assumption that  $\pi$  is a random oracle (secure hash function) from  $G \times G$  to  $Z_{2^{\lceil \log q \rceil / 2}}^*$ . Since the computational cost for Alice is the same as that for Bob. In the following, we will only analyze Alice's computation.

First, Alice needs to choose a random number  $x$  and compute  $g_{\text{ID}_A}^x$  in the group  $G$ . In order for Alice to compute  $sk = \hat{e} \left( g_{\text{ID}_B}^{s_B} \cdot R_B, g_{\text{ID}_A}^{(x+s_A)\alpha} \right)$ , she needs to do 1.5 exponentiation in  $G$ , one multiplication in  $G$ , and one pairing. Thus in total, she needs to do 2.5 exponentiation in  $G$ , one multiplication in  $G$ , and one pairing.

Alternatively, Alice can compute the shared secret as  $sk = \hat{e} \left( g_{\text{ID}_B}^{s_B} \cdot R_B, g_{\text{ID}_A}^\alpha \right)^{(x+s_A)}$ . Thus for the entire IDAK protocol, Alice needs to do 1.5 exponentiation in  $G$  (one for  $g_{\text{ID}_A}^x$  and 0.5 for  $g_{\text{ID}_B}^{s_B}$ ), one multiplication in  $G$ , one pairing, and one exponentiation in  $G_1$ .

The IDAK protocol could be sped up by letting each participant do some pre-computation. For example, Alice can compute the values of  $g_{\text{ID}_A}^x$  and  $g_{\text{ID}_A}^{x\alpha}$  before the protocol session. During the IDAK session, Alice can compute the shared secret as  $sk = \hat{e} \left( g_{\text{ID}_B}^{s_B} \cdot R_B, g_{\text{ID}_A}^{x\alpha} \cdot g_{\text{ID}_A}^{\alpha s_A} \right)$  which needs 1 exponentiation in  $G$  (0.5 for  $g_{\text{ID}_B}^{s_B}$  and 0.5 for  $g_{\text{ID}_A}^{\alpha s_A}$ ), 2 multiplications in  $G$ , and one pairing. Alternatively, Alice can compute the shared secret as  $sk = \hat{e} \left( g_{\text{ID}_B}^{s_B} \cdot R_B, g_{\text{ID}_A}^\alpha \right)^{x+s_A}$  which needs 0.5 exponentiation in  $G$ , one multiplication in  $G$ , one pairing, and one exponentiation in  $G_1$ . In a summary, Figure 1 lists the computational cost for Alice (an analysis of all other identity based key agreement protocols shows IDAK is the most efficient one, details will be given in the final version of this paper).

	without pre-computation		with pre-computation	
	choice 1	choice 2	choice 1	choice 2
pairing	1	1	1	1
exponentiation in $G$	2.5	1.5	1	0.5
multiplication in $G$	1	1	2	1
exponentiation in $G_1$	0	1	0	1

**Fig. 1.** IDAK Computational Cost for Alice